Simple and Robust Binary Self-Location Patterns

Alfred M. Bruckstein, Tuvi Etzion, *Fellow, IEEE*, Raja Giryes, Noam Gordon, Robert J. Holt, and Doron Shuldiner

Abstract—A simple method to generate a two-dimensional binary grid pattern, which allows for absolute and accurate self-location in a finite planar region, is proposed. The pattern encodes position information in a local way so that reading a small number of its black or white pixels at any place provides sufficient data from which the location can be decoded both efficiently and robustly.

Index Terms—de Bruijn sequences, M-sequences, self-location patterns

I. INTRODUCTION

TAKE a blindfolded man on a random one-hour walk around town and then remove his blindfold. How will he know where he is? He has several options, based on the information he can gather. The man could carefully count his steps and take note of every turn during the blindfolded walk to know his location relative to the beginning of his trip. Armed with a navigation tool such as a sextant or GPS unit, he could ask the stars or the GPS satellites where he is. Lastly, he could simply look around for a reference, such as a street sign, a landmark building, or even a city map with a little arrow saying "*You are here*."

There are numerous applications where a similar problem is encountered. We need to somehow measure the position of a mobile or movable device, using some sort of sensory input. Wheeled vehicles can count the turns of their wheels much like the man counting his steps. Similarly, many devices, from industrial machine stages to ball-mice, employ sensors which are coupled with the mechanics and count small physical steps of a known length, in one or more dimensions. The small relative position differences can be accumulated to achieve *relative self-location* to a known starting point. More recent technologies, such as those found in modern optical mice, use imaging sensors instead of mechanical encoders to estimate the relative motion by constantly inspecting the moving texture or pattern of the platform beneath them.

Sometimes the inherent accumulating error in relative self-location methods, or some other reasons, make them infeasible or unfit for certain applications, where we would want the

A. M. Bruckstein, T. Etzion, R. Giryes, N. Gordon, and D. Shuldiner are with the Department of Computer Science, Technion — Israel Institute of Technology, Haifa 32000, Israel. (email: freddy@cs.technion.ac.il, etzion@cs.technion.ac.il, raja@cs.technion.ac.il, ngordon@cs.technion.ac.il, doron.email@gmail.com).

R. J. Holt is with the Department of Mathematics and Computer Science, Queensborough, City University of New York, Bayside, NY 11364, USA. (email: RHolt@qcc.cuny.edu).

capability to obtain instant and accurate *absolute self-location*. Given several visible landmarks of known locations, a mobile robot could calculate its position through a triangulation [1]. Alternatively, cleverly designed space fiducials (e.g., [2]), whose appearance changes with the angle of observation, can also serve for self-location.

Much like street signs for people, there are absolute self-location methods that provide sufficient local information to the device sensors, such that the absolute positioning can be attained. Specifically, planar patterns have been suggested, where a small local sample from anywhere in the pattern provides sufficient information for decoding the absolute position. A naive example could consist of a floor filled with densely packed miniature markings, in which the exact coordinates are literally inscribed inside each marking. Of course, that would require a high sensor resolution and character recognition capabilities. Indeed, there are much more efficient methods, which do with considerably less geometric detail in the pattern. Some commercial products have been utilizing this approach, e.g., a pen with a small imaging device in its tip, writing on paper with a special pattern printed on it, which allows full tracking of the pen position at any time [3].

A classic method for absolute self-location in one dimension is the use of de Bruijn sequences [4], [5]. A de Bruijn sequence of order n over a given alphabet of size q is a cyclic sequence of length q^n, which has the property that each possible sequence of length n of the given alphabet appears in it as a consecutive subsequence exactly once. Thus, sampling n consecutive letters somewhere in the sequence is sufficient for perfect positioning of the sampled subsequence within the sequence. Several methods for the construction of de Bruijn sequences have been proposed, e.g. [5], [6], [7], [8]. There is also a two-dimensional generalization, i.e., it is possible to construct a two-dimensional cyclic arrays in which each rectangular sub-array of a certain size $k \times n$ appears exactly once in the array. These types of arrays are called *perfect maps*, e.g. [9], [10] and they can serve as the basis for absolute self-location on the plane.

Of special interest and importance in communication are maximal-length linear shift-register sequences known also as M-sequences or pseudo-noise sequences [11]. An *M-sequence* of order n is a sequence of length $2^n - 1$ generated by a linear feedback shift-register of length n. In a cyclic sequence of this type, each nonzero n-tuple appears exactly once as a window of length n in one period of the sequence exactly once. These sequences have many important and desired properties [11], [12]. A two-dimensional generalization of such sequences was presented in [12] and are called pseudo-random arrays. We note also that M-sequences can be used for robust one-dimensional location by using their error-correction properties,

as analyzed in [13].

In this work we propose a simple product construction to generate a two-dimensional binary patterns for absolute self-location. The paper is organized as follows. In Section II we present the product construction based on two sequences with some one-dimensional window properties. A two-dimensional array with optimal self-location based on sensing a cross shape is obtained by this construction. In Section III we prove that the same construction can be used for reasonably effective error-correction of self-location with a rectangular shape. Our conclusions and some interesting problems for future research are presented in Section III.

II. THE PROPOSED 2D SELF-LOCATION PATTERN

Our approach for building two-dimensional arrays with self-location properties is based on a product of two sequences, one of which is a de Bruijn sequence and the other being a sequence in which only half of the patterns appear.

A. Half de Bruijn Sequences

A *half de Bruijn sequence* of order n is a (cyclic) sequence of length 2^{n-1} which has the property that for each possible n-tuple X, either X or $\bar{X}$ (the bitwise complement of X), but not both, appear in the sequence exactly once as a subsequence.

There are many different ways to construct half de Bruijn sequences. One method, in which a half de Bruijn sequence of length 2^{n-1} is generated from a de Bruijn sequence of order $n-1$ by using the inverse of the well known mapping **D**, called the **D**-morphism, is described in [6]. Another one is based on M-sequences. The following results were proved in [7].

Theorem 1: If S is an M-sequence of order $n-1$ then for each pair of n-tuples X and $\bar{X}$ either X or $\bar{X}$ appears in S, with the exception of the pair which consists of the all-zero and all-one n-tuples.

Corollary 1: Let S be an M-sequence of order $n-1$ and let S' be the sequence obtained from S by adding another *one* to the unique run with $n-1$ *ones*. Then S' is a half de Bruijn sequence.

B. The Construction

For two sequences $\mathcal{T} = (t_1, \ldots, t_K)$ and $\mathcal{S} = (s_1, \ldots, s_N)$ the product $\mathcal{T} \otimes \mathcal{S}$ is a $K \times N$ array G in which $g_{ij}, 1 \leq i \leq K$, $1 \leq j \leq N$, contains the value $t_i \oplus s_j$ (where $\oplus$ denotes modulo 2 addition, also known as the XOR operator).

Take a half de Bruijn sequence $\mathcal{T} = (t_1, \ldots, t_K)$ and a de Bruijn sequence $\mathcal{S} = (s_1, \ldots, s_N)$ of orders k and n, and lengths $K = 2^{k-1}$ and $N = 2^n$, respectively, and let $G = \mathcal{T} \otimes \mathcal{S}$. Clearly, each row in G equals either $\mathcal{S}$ or $\bar{\mathcal{S}}$. Similarly, each column equals either $\mathcal{T}$ or $\bar{\mathcal{T}}$. Thus, each row and each column retain their window property and can serve for self-location in each dimension.

Theorem 2: Each cross shaped pattern with k vertical and n horizontal entries appears exactly once as a pattern in the array G.

Proof: Let X be a column vector of length k and Y be a row vector of length n. Either X or $\bar{X}$ appears in the

sequence $\mathcal{T}$. Let $\mathcal{X}$ the pattern which appears. Both Y and $\bar{Y}$ appear in the sequence $\mathcal{S}$. Crosses with vertical vector X and horizontal vector Y appear in G only in the portions related to $Z_1 \overset{\text{def}}{=} \mathcal{X} \otimes Y$ and $Z_2 \overset{\text{def}}{=} \mathcal{X} \otimes \bar{Y}$. Moreover, the crosses in Z_1 are complements of the crosses Z_2. For each cross inside Z_1 and Z_2 there are two possible assignments, depending on the mutual entry of the vertical and horizontal component. Each one of these values appears in either Z_1 or Z_2. ∎

By Theorem 2, we can use a cross sensor array to sample k vertical and n horizontal pixels (with one mutual pixel) in order to obtain self-location.

Corollary 2: The proposed method is optimal in terms of the number of sampled pixels required to achieve self-location with a cross of vertical length k and horizontal length n.

Corollary 3: In the array G each sampled sub-array of size $k \times n$ has a unique location.

Remark 1: Similar and more sophisticated product constructions to generate arrays with low redundancy and effective two-dimensional error-correction capabilities, were suggested in various papers, e.g. [14], [15].

Remark 2: In practice, the planar domain is generally not cyclic. In order to retain the ability to sense all $2^{k-1} \times 2^n$ possible locations with a sensor whose footprint is $k \times n$ pixels array, we extend $\mathcal{T}$ and $\mathcal{S}$ by appending their first $k-1$ and $n-1$ bits, respectively, to their ends. The result is now a $(2^{k-1} + k - 1) \times (2^n + n - 1)$ array.

Example 1: An example of our proposed two-dimensional grid pattern can be seen in Fig. 1. It was generated using a de Bruijn sequence of order 4 in the horizontal axis, and a half de Bruijn sequence of order 5 in the vertical axis, resulting in a cyclic array of 16×16 pixels. The first column and the first row in the figure contain the location indexes. The second column and the second row contain $\mathcal{T}$ and $\mathcal{S}$, respectively. From the bit values inside the grid we can decode our position. An example of a sensor readout is marked in the table. The sensor is a 5 by 4 cross. The vertical readout is 10010, and the horizontal readout is 1000 and its unique position can be easily decoded from $\mathcal{T}$ and $\mathcal{S}$.

C. Computing the Location

The first step in our method recovers the one-dimensional subsequences that correspond to the location in each dimension. Essentially, the two-dimensional problem is now reduced to two independent one-dimensional decoding problems. Decoding the location of a subsequence in a de Bruijn sequence is a well-known problem. Decoding of a half de Bruijn sequence is done similarly.

A classic approach of creating a de Bruijn sequence $\mathcal{S}$ of order n, requires $\mathbf{O}(n)$ space and $\mathbf{O}(n \cdot 2^n)$ time to generate the whole sequence $\mathcal{S}$ [5], [8]. This involves $\mathbf{O}(n)$ space and $\mathbf{O}(n \cdot 2^n)$ time, with n being the order of the de Bruijn sequence. If running time is an issue, one could create and store in advance a look-up table which lists the locations of all subsequences. This yields $\mathbf{O}(n)$ time complexity, but requires $\mathbf{O}(n \cdot 2^n)$ space for the table. For larger n, a more flexible trade-off between time and space complexity was suggested in [16]. A partial look-up table of evenly spaced locations

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
		0	0	0	0	1	1	1	1	0	1	1	0	0	1	0	1
1	1	1	1	1	1	0	0	0	0	1	0	0	1	1	0	1	0
2	1	1	1	1	1	0	0	0	0	1	0	0	1	1	0	1	0
3	1	1	1	1	1	0	0	0	0	1	0	0	1	1	0	1	0
4	1	1	1	1	1	0	0	0	0	1	0	0	1	1	0	1	0
5	1	1	1	1	1	0	0	0	0	1	0	0	1	1	0	1	0
6	0	0	0	0	0	1	1	1	1	0	1	1	0	0	1	0	1
7	1	1	1	1	1	0	0	0	0	1	0	0	1	1	0	1	0
8	0	0	0	0	0	①	1	1	1	0	1	1	0	0	1	0	1
9	1	1	1	1	1	⓪	0	0	0	1	0	0	1	1	0	1	0
10	1	1	1	1	①	⓪	⓪	⓪	0	1	0	0	1	1	0	1	0
11	0	0	0	0	0	①	1	1	1	0	1	1	0	0	1	0	1
12	1	1	1	1	1	⓪	0	0	0	1	0	0	1	1	0	1	0
13	1	1	1	1	1	0	0	0	0	1	0	0	1	1	0	1	0
14	1	1	1	1	1	0	0	0	0	1	0	0	1	1	0	1	0
15	0	0	0	0	0	1	1	1	1	0	1	1	0	0	1	0	1
16	0	0	0	0	0	1	1	1	1	0	1	1	0	0	1	0	1

Fig. 1. The 16×16 product array of $\mathcal{T} \otimes \mathcal{S}$. The marked cells illustrate a readout by a cross-shaped sensor.

called *milestones* is created in advance. During runtime, the algorithm which generates the sequence is initialized with the query subsequence and then iterated until one of the milestones is encountered. For example, this can yield $\mathbf{O}(n \cdot 2^{\frac{n}{2}})$ time complexity and will require $\mathbf{O}(n \cdot 2^{\frac{n}{2}})$ space for the table.

In either case, implementation of the self location process using modern computer systems is feasible, at least for reasonable and practical values of n, depending on the application. Take $n = 16$ for a concrete example. It allows a definition of 2^{16} locations, e.g., a resolution of 0.1mm over a range of about 6.5 meters. In the first approach it would take, in the worst case, about 65k simple iterations (on a 16-bit register), which can be performed reasonably quickly on current modest embedded processors currently clocked at about tens or hundreds of Megahertz. In the second approach, the look-up table would consume about 128k bytes (each entry being a two-byte word), which is, again, a quite modest requirement given today's memory capabilities.

There are more efficient methods to generate de Bruijn sequences [17] which can be used in case of an application in which k and n are much larger. The problem of decoding perfect maps was considered for example in [18]. A comprehensive survey on this topic was given in [19].

III. ROBUST SELF-LOCATION

The cross shaped sensor is rather 'spread out', so it might be a disadvantage in applications. In this section we show that this weakness becomes an advantage for robust self-location when the sensor is of a rectangular shape. If we use a $k \times n$ pixel sensor (see Corollary 3), we can utilize the inherent redundancy within the kn bits to decode the location while overcoming a considerable number of faulty bits readings. This is also a very practical choice, considering that two-dimensional rectangular sensor grids are the most common variety and are the standard choice for most applications.

We assume that less than quarter of the bits in each row and less than half of the bits in each column of the input array are in error. As it will be shown in the sequel, this is a fair assumption which can account for quite strong noise in practical terms. The algorithm for robust self-location presented in Fig. 2 is a simple majority decoding.

Robust self-location algorithm

The algorithm's input is a rectangle $Z \overset{\text{def}}{=} \{z_{ij} \ : \ 1 \le i \le k, 1 \le j \le n\} = (X \otimes Y) \oplus \mathcal{E}$; where X is a vertical k-tuple of a vertical half de Bruijn sequence $\mathcal{T}$; Y is a horizontal n-tuple of a horizontal de Bruijn subsequence $\mathcal{S}$; and $\mathcal{E}$ is a $k \times n$ error pattern. We assume that less than $\frac{n}{4}$ of the bits in each row of $\mathcal{E}$ are *ones* and less than $\frac{k}{2}$ of the bits in each column of $\mathcal{E}$ are *ones*. The output is the original horizontal and vertical subsequences X and Y, respectively.

- Assume that the first bit of X is b. Let D be the first row of Z.
- For each row A of Z
  - if more than half of the bits of $A \oplus D$ are *zeroes* then the corresponding bit of X is b
  - otherwise, the corresponding bit of X is $\bar{b}$.
- Assign 0 or 1 to b to obtain X which appears in $\mathcal{S}$.
- For each column B of Z
  - if more than half of the bits of $B \oplus X$ are *zeroes* then the corresponding bit in Y is a *zero*
  - otherwise, the corresponding bit in Y is an *one*.

Fig. 2. The robust self-location algorithm.

Theorem 3: Given a grid of size $2^{k-1} \times 2^n$ and a $k \times n$ pixel sensor, if less than quarter of the bits in each row and less than half of the bits in each column are in error, then the algorithm accurately decodes the sensor location.

Proof: Since the number of errors in a row is less than $\frac{n}{4}$ it follows that two rows which were originally the same will agree in more than half of their bits and two complement rows will disagree in more than half of their bits. Therefore, the related bits of X will be the same or different, respectively. Having all the k bits of X in terms of the variable b, there is only one assignment of a legal k-tuple since the vertical sequence is a half de Bruijn sequence.

Having the correct vertical subsequence X and since the number of errors in a column is less than $\frac{n}{2}$ it follows that if X agree in more than $\frac{n}{2}$ bits with a column then the corresponding bit of Y is a *zero*; and if it disagree in more than $\frac{n}{2}$ bits with a column then the corresponding bit of Y is

a *one*.

Remark 3: Decoding can be done also if more than quarter of the bits in some rows are in error. A slightly better condition would be to require that the number of distinct positions in error in any two rows is less than $\frac{n}{2}$. This requirement can be further improved.

Similar algorithm will also work if we will exchange between rows and columns, or equivalently if we will consider a transposed array. Therefore, we can exchange our assumption on the number of wrong bits in a row or a column. But, having for example at least half of the bits wrong in a given column (or a given row) will cause a wrong identification of the original subsequences.

Lemma 4: Given a grid of size $2^{k-1} \times 2^n$ and a $k \times n$ pixel sensor, if at least half of the bits in one of the columns of a pixel sensor are in error, then we cannot ensure accurate decoding of the original subsequences.

Proof: Let X and Y two n-tuples which differ only in the first bit. Both X and Y appears as a window of length n in the de Bruijn sequence $\mathcal{S}$ of order n. Let Z be a k-tuple which appears as a window in the sequence $\mathcal{T}$. The products $Z \otimes X$ and $Z \otimes Y$ appears as $k \times n$ windows in the array $\mathcal{T} \otimes \mathcal{S}$. Both $k \times n$ windows differ only in the first column and it would be impossible to distinguish between the two windows if half of the bits in the first column are in error. If more than half of the bits in the first column are in error then a wrong decoding of the sensor location will be made. The same arguments can be applied to any other column. ∎

We note that by Lemma 4 we cannot correct $\left\lceil \frac{k}{2} \right\rceil$ or more random errors in a $k \times n$ array. The reason is that the array is highly redundant. This is quite weak from an error-correction point of view. But, by Theorem 3 we are able to correct about $\frac{kn}{4}$ errors in an $k \times n$ array if less than $\frac{n}{4}$ errors occur in a row and less than $\frac{k}{2}$ errors occur in a column. The reason is that redundant rows and columns are used for the majority decoding. This result is quite strong from error-correction point of view. Thus, the weakness for one type of errors becomes an advantage for another type of errors.

Example 2: The 7×9 sub-array of Fig. 3 has no more than two errors in a row and three errors in a column. The first row has more than half bits in common with the 5th and the 7th rows. Thus the vertical pattern is $b\bar{b}\bar{b}\bar{b}\bar{b}b$. Suppose that $b = 1$, i.e. the vertical pattern is 1000101. We now compare all of the columns with 1000101. If more than half of the corresponding bits agree, the bit in the horizontal sequence is *one*; otherwise it is a *zero*. Thus, the sequence is 110111001. The sub-array with no errors is presented in Fig. 4.

Now, we analyze the error rates in individual bits that allow us to determine the probability that the position is determined correctly. Given a $k \times n$ rectangle in which the probability of each bit being correct is p independent of the other bits, we can determine the probability that each row satisfies the condition above, that less than quarter of the bits are in error. Then the probability that each row is satisfactory is the individual row probability raised to the kth power, the number of rows. For simplicity we assume now that $k = n$.

To find the probability that all of the rows satisfy the row condition, we raise the probabilities $P(n; p)$ to the power of

$$
\begin{array}{ccccccccc}
1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
\end{array}
$$

Fig. 3. A 7×9 sub-array with errors.

$$
\begin{array}{ccccccccc}
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
\end{array}
$$

Fig. 4. The corrected sub-array.

the number of rows. These are given in Table I.

$p \setminus n$	8	16	32	64
0.90	0.191	0.322	0.687	0.9858
0.91	0.253	0.443	0.817	0.9957
0.92	0.329	0.573	0.906	0.9989
0.93	0.417	0.699	0.959	0.9998
0.94	0.517	0.809	0.985	> 0.9999
0.95	0.624	0.894	0.996	> 0.9999
0.96	0.733	0.951	0.9991	> 0.9999
0.97	0.835	0.982	0.9999	> 0.9999
0.98	0.920	0.9962	> 0.9999	> 0.9999
0.99	0.979	0.9997	> 0.9999	> 0.9999

TABLE I

PROBABILITIES THAT EACH OF THE ROWS OF AN $n \times n$ WINDOW HAS FEWER THAN QUARTER OF ITS BITS IN ERROR WHEN THE PROBABILITY THAT EACH BIT IS CORRECT IS p.

In order to have a probability of at least 0.99 that the row condition is satisfied, we need $p > 0.994$ for $n = 8$, $p > 0.98$ for $n = 16$, $p > 0.95$ for $n = 32$, and $p > 0.91$ for $n = 64$. In order for the row condition to be satisfied with probability at least 0.999, it is sufficient that $p > 0.99$ for $n = 16$, $p > 0.96$ for $n = 32$, and $p > 0.93$ for $n = 64$. Also, if $p > 0.98$ for $n = 32$ or $p > 0.94$ for $n = 64$, the row condition is satisfied with probability greater than 0.9999 .

The probability that the column condition (that less than half the bits are in error) is not satisfied when the row condition is satisfied is negligible. For example, if we let $Q(n; p)$ represent the probability that the column condition is not satisfied, i.e.

$$
Q(n; p) = \sum_{i=\lceil \frac{n}{2} \rceil}^{n} \binom{n}{i} p^i (1-p)^{n-i} , \qquad (1)
$$

then we have results such as $Q(16; 0.99) = 1.2 \times 10^{-12}$. In a square array, the probability that the column condition is not satisfied for at least one of the columns is then $1 - (1 - 1.2 \times 10^{-12})^{16} = 1.9 \times 10^{-11}$.

IV. CONCLUSION AND FUTURE RESEARCH

Implementing absolute self-location in a planar region using special patterns is a viable and proven approach and can solve a variety of technological problems. In this paper we proposed a solution based on robust two-dimensional arrays with a two-dimensional window property. The method also has a rather strong error-correction capability. It enables to correct errors

if less than quarter of the bits in a row and less than half of the bits in a column are in error.

In some applications, the alignment of the sensor array to the grid pattern is not guaranteed. The sensor may be arbitrarily translated and rotated, so that retrieving the local bit matrix is not trivial. Position location of one-dimensional sequences, when the orientation of the subsequence is not known was considered in [20]. The solution in two-dimensional arrays is to sample the region at a somewhat higher resolution than k by n, and analyze the image in order to first estimate the pose of the pattern of rows and columns. Since the proposed pattern has a very pronounced structure consisting of identical or inverted rows (as well as columns), this can greatly aid in the task. Using an M-sequence and its complement as vertical and horizontal sequences in our construction can also help in solving of the orientation problem. A complete analysis of these issues is a problem for future research.

There are many other future research problems in this area. Some related to our specific construction and some are to new possible construction methods.

1) As indicated in Remark 3, the claim in Theorem 3 can be strengthened. What is the strongest claim on the error capability of our scheme? Do the de Bruijn sequence and the half de Bruijn sequence that we selected have any influence on this claim?

2) How can we improve the error-correction capabilities of our scheme if the de Bruijn sequence and the half de Bruijn sequence are derived from M-sequences with error-correction capabilities as indicated in [13].

3) The array obtained by our method can correct a limited number of random errors, even so we proved that the probability for such errors which the method cannot correct is negligible. Generating arrays with window properties which can correct large number of random errors is an important topic for future research.

4) Finally, we note that a folding method for generating pseudo-random arrays from M-sequences was suggested in [12]. This method was subsequently generalized in [21]. Can this method be adapted also to generate better pseudo-random arrays which can correct random errors? Using the M-sequences as suggested by [13] for this purpose could be the first step in attempting to find an answer to such questions..

REFERENCES

[1] C. Cohen, F. V. Koss, "A Comprehensive Study of Three Object Triangulation," *Proceedings of the 1993 SPIE Conference on Mobile Robots*, pp. 95–106, 1992.

[2] A. M. Bruckstein, R.J. Holt, T.S. Huang and A.N. Netravali, "New devices for 3d pose estimation: Mantis eyes, Agam paintings, sundials, and other space fiducials." *International Journal of Computer Vision*, vol.39/2, pp. 131–140, 2000.

[3] Anoto Group AB website, http://www.anoto.com.

[4] N. G. de Bruijn, "A Combinatorial Problem." *Koninklijke Nederlandse Akademie v. Wetenschappen*, vol. 49, pp. 758–764, 1946.

[5] H. Fredricksen, "A survey of full length nonlinear shift register cycle algorithms," *SIAM Review*, vol.24, pp. 195–221, April 1982.

[6] A. Lempel, "On a Homomorphism of the de Bruijn Graph and its Applications to the Design of Feedback Shift Registers," *IEEE Transactions on Computers*, vol. 19, pp. 1204–1209, December 1970.

[7] B. Arazi, "Method of constructiong de Bruijn sequences," *Electronic letters*, vol. 12, pp. 858–859, December 1976.

[8] T. Etzion, A. Lempel, "Algorithms for the generation of full-length shift-register sequences," *IEEE Transactions on Information Theory*, vol. 30, pp. 480–484, May 1984.

[9] T. Etzion, "Constructions for perfect maps and pseudorandom arrays," *IEEE Transactions on Information Theory*, vol.34, pp.1308–1316, September 1988.

[10] K. G. Paterson, "Perfect maps", *IEEE Trans. Inform. Theory*, vol. 40, pp. 743–753, May 1994.

[11] S. W. Golomb, *Shift Register Sequences*, Aegean Park Press, 1982.

[12] F. J. MacWilliams, N. J. A. Sloane, "Pseudo-random sequences and arrays", *Proc. IEEE*, vol. 64, pp. 1715–1729, December 1976.

[13] P. V. Kumar and V. K. Wei, "Minimum distance of logarithmic and fractional partial m-sequences," *IEEE Transactions on Information Theory*, vol. 38, pp. 1474–1482, September 1992.

[14] M. Breitbach, M. Bossert, V. Zyablov, and V. Sidorenko, "Array codes correcting a two-dimensional cluster of errors", *IEEE Trans. on Inform. Theory*, vol. 44, pp. 2025–2031, September 1998.

[15] T. Etzion and E. Yaakobi, "Error-Correction of Multidimensional Bursts", *IEEE Trans. on Inform. Theory*, vol. 55, pp. 961–976, March 2009.

[16] E. Petriu, "New pseudorandom/natural code conversion method", *Electronics Letters*, vol.24, pp.1358–1359, October 1988.

[17] C.J. Mitchell, T. Etzion, K.G. Paterson, "A method for constructing decodable de Bruijn sequences," *IEEE Transactions on Information Theory*, vol. 42, pp. 1472–1478, September 1996.

[18] C. J. Mitchell and K. G. Paterson, "Decoding perfect maps", *Designs, Codes, and Cryptography*, vol. 4, pp. 11–30, 1994.

[19] J. Burns and C. J. Mitchell, "Coding schemes for two-dimensional position sensing", in M. J. Ganley sed., *Cryptography and Coding III: Proceedings of the third IMA Cryptography and Coding Conference*, Cirencester, December 1991, Oxford University Press, pp. 31–66, Oxford, 1993.

[20] Z. D. Dai, K. M. Martin, M. J. B. Robshaw, and P. R. Wild, "Orientable sequences", in M. J. Ganley sed., *Cryptography and Coding III: Proceedings of the third IMA Cryptography and Coding Conference*, Cirencester, December 1991, Oxford University Press, pp. 31–66, Oxford, 1993.

[21] T. Etzion, "Sequence folding, lattice tiling, and multidimensional coding", *IEEE Trans. on Inform. Theory*, vol.57, pp. 4383–4400, July 2011.